Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 18 (2013) 2607 – 2610

Procedia Computer Science

International Conference on Computational Science 2013

MataNui – A Distributed Storage Infrastructure for Scientific Data

Guy K. Kloss

School of Computing + Mathematical Sciences,
Auckland University of Technology, Auckland, New Zealand

Abstract

In science and engineering the problem of "sanity" in data management is quite common. Particularly, if partners within a project are geographically distributed, and require access to this data. These partners would ideally like to access or store data using a local server, while still retaining remote access. Furthermore, data does not "live alone," but has a "partner": Meta-data. Currently available solutions cater for different needs, and do not fully address the problems in our own environment. Particularly, seamless integration within a Grid infrastructure is essential.

For these reasons, the presented *MataNui* data server infrastructure offers full storage of files along with arbitrarily extensive meta-data for each stored item. The server can replicate its content between geographically distributed locations, making each usable for full read/write access, enabling researchers to use the one closest to their own network. Files are accessible through GridFTP as the predominant protocol for Grid Computing. For full access to meta-data and server side search queries a (RESTful) Web Service is provided. The current implementation features very high performance in data throughput on storage, retrieval and query operations. Through the Web Service front end it is straight forward to integrate it with scientific workflow environments or scientific data management applications.

Keywords: Grid Computing, Data Fabric, big data, distributed, meta-data, GridFTP

1. Introduction

Scientists and engineers often work in teams, and shared data needs to be kept in a commonly accessible space. Difficulty for this increases with larger the teams, larger data and more extensive the geographical distribution. Data has to be accessible at all locations. To manage such data sensibly, arbitrary meta-data annotations are essential. One solution may be distributed file systems [1]. Other related developments, such as dCache and ExtreemFS do not feature the extensive meta-data capabilities. iRODS does, but it can be difficult to deploy and operate, and it requires third party services to expose itself to common Grid infrastructures. Lastly, Globus Online represents only a data transfer layer, but offers no storage or meta-data.

This paper presents *MataNui,* a solution for a file-based data server. It manages arbitrary amounts and structures of meta-data along with each data item. The storage engine can replicate data in distributed setups for access and storage at different (geographical) locations, ensuring fast access. To speed up finding items, it is able to

*Corresponding author. Tel.: +64-9-921-9999 ext. 5032.
E-mail address: guy.kloss@aut.ac.nz.

doi:10.1016/j.procs.2013.05.451

perform server-side queries on meta-data. Lastly, the system is intended to be robust, easy to deploy and easy to use. MataNui currently offers two different main access path ways: GridFTP and a generic Web Service. GridFTP is intended to interface seamlessly with common Globus Toolkit-based Grids, also adopting Grid certificate authentication. The RESTful JSON Web Service authenticates in the same way. It offers full access to all data and meta-data and allows for the server side query functionality.

In the following this paper first describes the underlying data-management concept in Sect. 2, followed by exposing the data through services in Sect. 3. Sect. 4 presents experiments used to judge the server's usability, with the results discussed in Sect. 5.

2. Concept

The key to the MataNui data management concept is a single data storage system operating geographically distributed, which is interfaced with appropriate server front-ends exposing the desired services or capabilities. This storage engine for files inclusive meta-data uses the NoSQL database *MongoDB* with its driver side file system implementation *GridFS*. MongoDB can perform sharding (horizontal partitioning) and replication (decentralised storage with cross-site synchronisation) "out of the box." Research teams can run local server instances (instead of remote access) to improve performance and latency. Such local storage also introduces redundancy, leading to better fault protection from server or networking issues. Each storage server can be exposed through different service front-ends as locally required, keeping a setup lean.

3. Service Implementation

MongoDB stores all file-based data and meta-data. Access services to it can be implemented in any language supporting a MongoDB client-side driver library, allowing for a platform, language or framework most suitable for the task. All components of the MataNui concept are licensed under open source licenses ensuring free usage or modification (except modifying MongoDB) within commercial or non-commercial environments.

GridFTP Server. Griffin [2] is a generic GridFTP server, implemented in Java using the Spring framework. Storage back-ends are provided through a plugin architecture. A new plugin has been implemented for the MataNui storage system. [3] description the concepts and implementation of Griffin in detail. In contrast to the previously available plugins (for iRODS or the local file system), MataNui was conceptualised as a native Grid component. Thus it is not necessary to perform mappings between a system's user ID and the *distinguished name* (DN) of Grid-based X.509 (proxy) certificates. The DN is used natively, indicating ownership.

Web Service. The MataNui Web Service implementation [4] provides additionally full access to *all* content (including custom meta-data), server side query execution, and versioning of stored files. It also uses Grid certificates for authentication (server *and* client side), and data ownership and access privileges are based on native Grid means. The Web Service is based on REST principles (using JSON encoding), and is easy to implement clients, or script it through command line HTTP clients (e. g GNU Wget, cURL). Technically, the Web Service is implemented as a *Python Web Server Gateway Interface* (WSGI) application running on an Apache Web server. To avoid unnecessary abstractions, it avoids higher level Web application frameworks, thus improving performant high volume data throughput by directly coupling the mod_wsgi streams with the PyMongo (Python MongoDB driver library) provided streams, which are both implemented in a native C library.

4. Experiments

The Griffin GridFTP server in general Grid deployments has been benchmarked extensively previously [3]. This paper analyses the suitability of MongoDB/GridFS as a storage system instead of the previously used iRODS system. Further experiments were conducted here to prove that the Web Service performs similarly. As a result of this paper and [3], MataNui then should also performs suitably in the less challenging distributed infrastructures (slower network, no client/server memory/disk competition).

The test system used separate hard disks to avoid competing I/O read/write operations on the same disk. MongoDB was configured to use a separate hard disk for storage. Where possible, "mock devices" were used (`/dev/null`, `/dev/zero` or fast random byte streams), outperforming physical file systems (> 400 MiB/s). For more specific information on the test setup see [5].

4.1. Boundaries and Preliminary Considerations

All tests have been conducted to eliminate bias as far as possible, and to be able to judge the performance of the server implementations (not network interface or client side file systems). Hard disk speed for both read/write operations were measured to all lie above 100 MiB/s, and all network operations were conducted using local host networking (> 40 Gbits/s or 5 GiB/s). Transfers into and out of MongoDB/GridFS storage were benchmarked for large file transfers to lie between 10 and 35 MiB/s. The great variance results from MongoDB's internal management using memory mapped files, thus heavily relying on available system memory. Therefore, it is also strongly influenced by the operating system's virtual memory management. Refer to [5] on more detailed information for these measurements. These preliminary considerations result in a ceiling for non-cached throughput of about 35 MiB/s to and from MongoDB. This equates to a theoretical network performance of 280 Gbit/s for the tested setup. Therefore, values close to this in Sect. 5 can be assume to have been limited by the infrastructure rather than the server implementation. Values above are a result of caching effects (e. g. repeated reads or writes absorbed by currently unused memory).

4.2. GridFTP Tests

GridFTP is an encrypted protocol, and the SSL layer may optionally use compression on transmitted data packages. To avoid measurement fluctuation, not compressible or random data were used in tests. Any retrieved output was directed to `/dev/null` to avoid local disk writes. All GridFTP transfers were conducted using `globus-url-copy`. Each invocation of this client involves a significant overhead establishing the secured connection, data and control channels. Therefore, tests with individual command invocations are only permissible on a few, large transfers. To reduce this overhead in transfers of many small files, `globus-url-copy` was invoked with a wild-card specification, using multiple data channels concurrently (with the `-cc N` option).

4.3. Web Service Tests

For the same reasons as above, fast generated random data streams were used on the client side, and retrieved file data was discarded (not written to file system). All tests against the RESTful Web Service were conducted using the cURL Python bindings. To accommodate concurrency in transfers and avoid "GIL problems," several Python "workers" (independent processes) were invoked to transmit individual files. The Python WSGI application on the server was configured to run two processes allowing for five threads each.

5. Results

Due to MongoDB's heavy usage of RAM, some interference between the database, server front-end (GridFTP or Web Service server) and client application is expected. Thus, individual runs on transfers of large files have exposed large variations in individual throughput (> 50 %), and results were averaged over four runs.

To estimate the maximum throughput, large files (sized at order of magnitude of machine's RAM) were transferred, reducing the impact of connection overhead and caching effects. Files of 1.5, 3.0 and 6.0 GiB were written/read multiple times to/from the MataNui servers. Measurement data (see Tab. 1) shows to that the maximum throughput rates lie around the estimated ceiling performance (35 MiB/s, see Sect. 4.1). Accessing the Web Service may be slightly more efficient than the GridFTP protocol, but this cannot be confirmed due to measuring uncertainty. Some values in testing the Web Service had to be omitted due to errors on concurrent transfers (the cURL library in the client application was consuming too much memory).

The use case of a large number of small files has been equally benchmarked (storing/retrieving content is very short compared to the connection overhead). A set of 7383 compressed files was used (median size 400 KiB, data volume 2.9 GiB). To avoid invoking `globus-url-copy` for each file, a wild card expression was used to store local files on the server, transferring files sequentially (similar behaviour to the Web Service client).

Table 1. Throughput results for data files in MiB/s. (Estimated error of measurements: approx. ± 15 % for large and ± 10 % for many files.)

		1.5 GiB		3.0 GiB		6.0 GiB		many individual files		
Server	concurrency	write	read	write	read	write	read	concurrency	write	read
GridFTP	1	32	29	24	27	27	30	1	2.2	–
GridFTP	2	23	24	19	20	23	22	4	4.6	–
GridFTP	4	17	19	17	16	24	24	16	4.5	–
Web Service	1	40	41	40	41	39	39	1	3.3	3.6
Web Service	2	20	42	20	–	19	–	4	10.1	13.5
Web Service	4	–	–	–	–	–	–	16	16.6	25.0

As expected, small file transfers perform significantly worse by volume than large file transfers. Increased data channel concurrency improved both the GridFTP as well as the Web Service transfers, but GridFTP incurs a higher transfer overhead than the Web Service. Within the tested cases, the Web Service seems to scale more linearly in read/write operations, whereas the GridFTP transfers already "flattens out" at four concurrent channels. Values missing for this test case are due to the Griffin server not yet supporting wild card expressions for server side files.

Additionally to file content transferred, a set of 61 key–value pairs of meta-data is associated with each of the 7383 files. These were added to the stored files through separate Web Service requests (to not bias throughput measurements). These individually very fast operations have not been timed for the purpose of this paper, but the majority of the time for these requests lies in establishing individual HTTPS connections.

6. Conclusion and Future Work

This paper describes the development of MataNui, a Grid compatible data storage system for file based data. The goal is to provide fast and efficient distributed file storage. It uses the NoSQL database MongoDB with GridFS to store data, allowing for arbitrary meta-data on each file, replication (decentralised storage with cross-site synchronisation) and horizontal partitioning through sharding (for scalability). It is exposed to the outside via a RESTful Web Service and the Griffin GridFTP server, both employing native Grid paradigms and authentication mechanisms. This allows for direct use with institutional Shibboleth and *short lived certificate service* (SLCS).

MataNui can already compete with distributed storage alternatives (e. g. iRODS) based on most used features. The MataNui infrastructure is simple to deploy and configure, while still offering good performance. This is achieved by building on common reliable components available in general Linux distributions.

The MataNui RESTful Web Service will soon be ported to Python 3.x and enhanced by further search capabilities, removing the need for extensive client-server interactions. Group access control will be added through use of virtual organisations (VOs). On the client side, the graphical scientific data management application DataFinder [6, 7, 8] will interfaced. Additionally, a FUSE file system implementation is planned to integrate the RESTful Web Service seamlessly into Linux and Mac OS operating systems.

References

[1] J. Darcy, Introduction to Distributed Filesystems, `http://pl.atyp.us/wordpress/index.php/2011/02/introduction-to-distributed-filesystems/`, last accessed March 2013 (February 2011).
[2] S. Zhang, G. K. Kloss, L. Behnke, Griffin Project, `https://code.google.com/p/datafabric-griffin/`, last accessed March 2013.
[3] S. Zhang, P. Coddington, A. Wendelborn, Connecting arbitrary data resources to the Grid, in: Proceedings of the 11th International Conference on Grid Computing (Grid 2010), ACM/IEEE, Brussels, Belgium, 2010.
[4] G. K. Kloss, MataNui Project, `http://launchpad.net/matanui`, last accessed March 2013.
[5] G. K. Kloss, MataNui Concept and Performance Measurement Environment, last accessed March 2013 (March 2013). doi:10.6084/m9.figshare.643855.
[6] T. Schlauch, A. Schreiber, DataFinder – A Scientific Data Management Solution, in: Proceedings of Symposium for Ensuring Long-Term Preservation and Adding Value to Scientific and Technical Data 2007 (PV 2007), Munich, Germany, 2007.
[7] DataFinder Project, `http://launchpad.net/datafinder`, last accessed March 2013.
[8] M. Ney, G. K. Kloss, A. Schreiber, Using Provenance to support Good Laboratory Practice in Grid Environments, Data Provenance and Data Management in eScience 426 (2013) 157–180. doi:10.1007/978-3-642-29931-5.